2.23 Verification Results for Moving Target Indicator

No major errors or discrepancies were found for MTI in ALARM 3.0. Minor problems were found in the input-checking routine (RDRERR), and one error was found in the input guide of the User's Manual. Code quality is good, and only minor problems with internal documentation were found. A summary of information for each design element is presented in the table below. A check mark indicates no discrepancies were found for a specific design element.

Table 2.23-1 Verification Results Summary

DESIGN ELEMENT	CODE LOCATION	DESK CHECK RESULT	TEST CASE ID	TEST CASE RESULT
23-1 Multiple Delay, Single PRF Response	RESMTI, 116-127	✓	23-3	✓
23-2 Single Delay, Staggered PRF Response	RESMTI, 128-148	✓	23-1	✓
23-3 Double Delay, Staggered PRF Response	RESMTI, 149-175	✓	23-2	✓
23-4 Multiple Delay, Staggered PRF Response	RESMTI, 116-127	✓	23-3	✓
23-5 Clutter Spectrum	CLTMTI, 103-111	✓	23-4	✓
23-6 Clutter Attenuation Factor	PULSED,315-323 ROMMTI, All CLTMTI, All	✓	23-4,5	✓
23-7 MTI Gates	MTIGAT, All	✓	23-6	✓
23-8 Noise and Noise Jamming Attenuation	RDRINT, 534-538	✓	23-7	✓
23-9 Algorithm Selection	PULSED 323,333,442 ROTPUL 140-150 SYSNOI 330-350 JAMMER 385-405, 645-655	✓	23-9 through 23-13	✓
23-10 Response Limiting	RESMTI, 182	✓	23-8	✓
Input	RDRINP, 245-248 266-269, 278-296	✓	23-14,15	✓
Echo Input	RDRPRT 241-256, 322-331 372-451	✓	23-14,15	✓
Error Checks	RDRERR 174-177, 225-228 408-411, 502-563	✓	23-14,15	23-15

2.23.1 Overview

The doppler frequency shift produced by a moving target may be used in a pulse radar to discern moving from fixed targets. Both Moving Target Indicator (MTI) and pulse doppler radars extract the doppler frequency shift for this purpose. Although MTI is a general term that applies to any pulse radar system that extracts and makes use of doppler information, it is used here to refer to those systems characterized by phase detectors using delay-line cancellation.

Implementation of the MTI functional element is controlled by subroutine PULSED. Subroutines ROMMTI, CLTMTI, MTIGAT, and RESMTI are called by PULSED specifically to perform MTI processing. PULSED also calls subroutines TARGPU, ROTPUL, and JAMMER to handle calculations (including MTI) for target body, target rotor, and jammer signals. Subroutines RDRINP, RDRERR, RDRPRT, and RDRINT are used for input data handling. Details of software design are described in Section 2.23.3 of ASP II.

2.23.2 Verification Design Elements

A design element is an algorithm that represents a specific component of the FE design. The ten design elements for MTI are described in detail in Section 2.23.2 of the ASP II and are summarized below in Table 2.23-2

.

Table 2.23-2 Design Elements

SUBROUTINE	DESIGN ELEMENT	DESCRIPTION
RESMTI	23-1 Multiple Delay, Single PRF Response	Compute the power response of a multiple-stage delay-line canceller for a single PRF system.
RESMTI	23-2 Single Delay, Staggered PRF Response	Compute the normalized power transfer function for the frequency response using staggered PRFs and one delay stage.
RESMTI	23-3 Double Delay, Staggered PRF Response	Compute the normalized power transfer response using staggered PRFs and two delay stages.
RESMTI	23-4 Multiple Delay, Staggered PRF Response	Compute the power response of a multiple-stage delay-line canceller using staggered PRFs.
CLTMTI	23-5 Clutter Spectrum	Compute the clutter spectrum.
PULSED ROMMT CLTMTI	23-6 Clutter Attenuation Factor	Compute the MTI attenuation factor for clutter.
MTIGAT	23-7 MTI Gates	Determine if the target is within an MTI gate.
RDRINT	23-8 Noise and Noise Jamming Attenuation	Calculate the MTI attenuation factor for receiver noise and noise jamming as the average gain of the MTI filter.
PULSED ROTPUL SYSNOI JAMMER	23-9 Algorithm Selection	Select and apply the correct MTI attenuation factor computation.
RESMTI	23-10 Response Limiting	Limit the MTI attenuation factor based on user input.
RDRINP	Input	Read user inputs for MTI in DATARDR
RDRPRT	Echo Inputs	Print values of user inputs
RDRERR	Error Checks	Check user inputs to ensure they are within appropriate limits

2.23.3 Desk Checking Activities and Results

The code implementing this FE was manually examined using the procedures described in Section 1.1 of this report. No discrepancies were found and the code quality was good.

The overall usage of headers and internal code documentation for the subroutines implementing MTI is quite good. Descriptive comments are interspersed throughout the code at regular intervals; these give the reader a good understanding of the processes involved in a module. Some header information categories are missing from several subroutines:

CLTMTI lacks purpose, inputs, and outputs sections
CLUTPU lacks purpose and inputs sections
PULSED lacks a purpose section
ROMMTI lacks purpose, input, and output sections
TARGPU lacks a purpose section

Many spelling errors were observed in the code comments.

Logical flow and subroutine inputs and outputs were examined and found to match descriptions in the ASP II. The ALARM 3.0 manuals were examined and found to be good documentation for the MTI functional element.

2.23.4 Software Test Cases and Results

Software testing was performed in most cases by running the entire ALARM model in debug mode. For these tests, ALARM was run in contour mode using the input data files for Sample 11 that were delivered with the ALARM code. These data files were modified by changing the contour step size from 500m to 5,000m to avoid lengthy run times. Any other data modifications or run procedures are listed in each test description. In Sample 11, the number of delay line cancellers (NDELAY) is 2 and the number of PRFs (NPRFS) is 2.

MTI testing included verifying initialization of specific input parameters and checks on intermediate calculations. Furthermore, all lines of code related to MTI were executed by reaching each logical branch. Table 2.23-3 describes the MTI tests. Design Element 23-1 is the most simple case of Design Element 23-4, where the number of PRFs equals one. Thus, the test of Design Element 23-1 is embedded in the test of Design Element 23-4.

Test cases 23-8 through 23-13 were designed to test that the correct formulas were accessed to calculate the MTI attenuation factor for various signals. Some of these tests were combined when they were actually performed. They are separated here for clarity and ease of explanation, even though this does lead to repetition in the test descriptions. Test cases 14 and 15 checked the reading, echoing and bounds checking of the user inputs for MTI.

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-1	OBJECTIVE: Test MTI Response for Single Delay, Staggered PRF. PROCEDURE: 1. In Datablock DATARADR, set: a IRADAR = 1 b NDELAY = 1 c. NPRFS = 3 d. PRFHZ(3) = 732 2. Run ALARM. 3. Record value of IRADR, NDELAY, NPRFS, X, FREQ, PIOPRF(NPRFS), SUMRES, RESTMP. 4. Independently calculate values of variables in step 3. VERIFY: Values in step 3 match those in step 4. RESULT: OK
23-2	OBJECTIVE: Test MTI Response for Double Delay, Staggered PRF. PROCEDURE: 1. In Datablock DATARADR, set: a. IRADAR = 1 b. NDELAY = 2 c. NPRFS = 3 d. PRFHZ(3) = 732 2. Run ALARM. 3. Record value of X, FREQ, PIOPRF(1), PIOPRF(NPRFS), SUMRES, RESPRF, RESTMP. 4. Independently calculate values of variable in step 3. VERIFY: Values in step 3 match those in step 4. RESULT: OK
23-3	OBJECTIVE: Test MTI Response for Multiple Delay, Staggered PRF. PROCEDURE: 1. In Datablock DATARADR, set: a. IRADAR = 1 b. NDELAY = 3 c. NPRFS = 3 d. PRFHZ(3) = 732 2. Run ALARM. 3. Record value of X, FREQ, PIOPRF(NPRFS), SUMRES, RESPRF, RESTMP. 4. Independently calculate values of variables in step 3. VERIFY: Values in step 3 match those in step 4. RESULT: OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-4	OBJECTIVE: Test computation of Clutter Spectrum and Clutter Signal Attenuation Factor. PROCEDURE: 1. In Datablock DATARADR, set: a. CLCNST = 1E-6 b. CONSRT = 0 c. SIGMAC = 10 2. Run ALARM. 3. In subroutine PULSED, record the value of SIGMA3, FLOW, FMID, and FHIGH. Compare with independently calculated values. 4. In subroutine ROMMTI, record the values of AX, BX, H, and HOVER2. a. Compare AX and BX with values of FLOW, FMID from PULSED. b. Compare H with user-defined tolerance EPS. c. Compare HOVER2 with independent calculation. 5. In subroutine ROMMTI, record the value of TRZOID(1). Compare with independent calculation. 6. In subroutine CLTMTI, record the initialization of CONSRT and SIGMA3. Compare to known values. 7. In subroutine CLTMTI, record the values of PREQ, CMXDFS, DELTAF, DELFSQ, TX, UX, and CLTMTI. Compare with independent calculation. 8. In subroutine PULSED, record the values of DPART1, DPART2, and CLTFAC. Use DPART1 and DPART2 to calculate CLTFAC independently. Compare the two values of CLTPAC. VERIFY: Compared values match in steps 3,4,5,6,7, and 8. RESULT: OK
23-5	OBJECTIVE: Test Romberg integration performed by subroutine ROMMTI. PROCEDURE: Use easily-defined geometric areas to test determination of area under a curve. 1. Rectangle. a. Define the function CLTMTI(X) = 5.0. b. Set A = 0.0 , B = 10.0 , EPS = 0.5. c. Run ROMMTI using an off-line driver. d. Record the value of the definite integral (variable DEFINIT). Compare with independent calculation. 2. Repeat step 1 using A = -10.0. 3. Repeat step 1 using A = 10.0 and B = -10.0. 4. Triangle. a. Define the function CLTMTI(X) = ABS(X). b. Set A = 0.0 , B = 10.0 , EPS = 0.5. c. Run ROMMTI using an off-line driver. d. Record the value of the definite integral (variable DEFINIT). Compare with independent calculation. 5. Repeat step 4 using A = -10.0. VERIFY: ROMMTI results match independent calculations. RESULT: OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-6	OBJECTIVE: Test determination of target position relative to MTI gates; i.e., check accuracy of logical variable MGATE. PROCEDURE: 1. In Datablock DATARADR, set four MTI gate azimuth and range values as follows: AZIMUTH gate minimum/maximum values, AMTIMN(1) = 20 AMTIMX(1) = 70 AMTIMN(2) = 100 AMTIMX(2) = 160 AMTIMN(3) = 200 AMTIMX(3) = 250 AMTIMN(4) = 280 AMTIMX(4) = 360 RANGE gate minimum/maximum values, RMTIMN(1) = 10 RMTIMX(1) = 20 RMTIMN(2) = 20 RMTIMX(2) = 35 RMTIMN(3) = 35 RMTIMX(3) = 40 RMTIMN(4) = 40 RMTIMX(4) = 50 2. Insert WRITE statements in subroutines PULSED and MTIGAT to print values of flight path number, azimuth and range, and logical variable MGATE. 3. Run ALARM. 4. Independently determine the correct value of MGATE (based on the gate definitions in step 1) for each flight path position. VERIFY: Value of MGATE in step 3 matches value in step 4. RESULT: OK
23-7	OBJECTIVE: Test calculation of average gain of MTI filter. PROCEDURE: 1. In Datablock DATARADR, set NDELAY = 0. 2. Run ALARM. 3. Note value of AVGMTI, and compare with the correct value (1.0). 4. In Datablock DATARADR, set NDELAY = 3. 5. Run ALARM. 6. Note value of AVGMTI, and compare with independent calculation. VERIFY: ALARM values of AVGMTI match independent calculations. RESULT: OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-8	OBJECTIVE: Test MTI Response limitation. PROCEDURE: 1. In Datablock DATARADR, set: a. IRADAR = 1 b. NDELAY = 3 c. NPRFS = 3 d. PRFHZ(3) = 732 e. FMTIDB = 40 2. Run ALARM. 3. Note value of variable GMNMTI in subroutines RDRINT and RESMTI. Compare to independent calculation. 4. Note values of RESTMP, GMNMTI, RESOUT, RESMTI. Compare with independent calculations. VERIFY: ALARM values match independent calculations in steps 3 and 4. RESULT: OK
23-9	OBJECTIVE: Check that subroutine PULSED uses the MTI clutter attenuation factor correctly. PROCEDURE: 1. In Sample 11, change DXYPLOT to 30,000. Run ALARM using VMS debug with this revised Sample 11 as input. 2. Set a breakpoint at the beginning of PULSED and step through this routine. 3. At line 323, note the value of CLTFAC. 4. At line 333, note value of MGATE, then note value of CLTATN at line 442. 5. Continue execution, setting a watch to observe when MGATE changes value. When this happens, note the value of CLTATN at line 442. VERIFY: When MGATE = False, CLTATN = 1; when MGATE = True, CLTATN = CLTFAC. RESULT OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-10	OBJECTIVE: Check that subroutine ROTPUL correctly calculates target rotor signal MTI attenuation factor. PROCEDURE: 1. Add a DATAROTO block to the revised Sample 11 input file of Test 23-8 with IPRROT = 0 IBSQM = 1 NPSPECT = 2 DOPKHZ = 100, 150 RCSINP = 1, 10 2. Run ALARM using VMS debug with the revised Sample 11 input file, setting breakpoints in PULSED, ROTPUL, and RESMTI. 3. Note the value of MGATE at line 333 in PULSED. Continue execution until MGATE = True. 4. Step through ROTPUL, comparing value of MGATE at line 140 to the PULSED value (True). 5. The first time through the loop, note the values of DOPCSR, DOPHZ(1), and FDOP at line 133. 6. Check whether line 142 or line 146 is executed in ROTPUL and note value of XMTI. 7. If RESMTI is called from ROTPUL, note value of the passed parameter (FDOP) in RESMTI. Compare with independently calculated value. 8. Note value of XMTI at end of ROTPUL. 9. Set a watch on MGATE to observe when it changes value in PULSED. 10. Repeat steps 6, 7, and 8. VERIFY: 1. Value of MGATE in ROTPUL matches its value in PULSED. 2. XMTI = 1 when MGATE is False and value of XMTI is calculated using RESMTI when MGATE is True. 3. DOPCSR matches independent calculations. RESULT: OK
23-11	OBJECTIVE: Check that subroutine SYSNOI calculates the system noise MTI factor correctly. PROCEDURE: 1. Run ALARM using VMS debug with the revised Sample 11 input file of Test 23-8. Set breakpoints in RDRINT, PULSED, and SYSNOI. 2. Note the value of AVGMTI at line 556 of RDRINT. 3. Note the value of MGATE at line 433 of PULSED. 4. Note the value of MGATET at line 339 in SYSNOI and compare to the value of MGATE in PULSED. 5. Check whether line 341 or 345 is executed in SYSNOI and note value of XMTINO. 6. Set a watch on MGATE, to observe when it changes value in PULSED. 7. Repeat steps 4 and 5. VERIFY: 1. The value of MGATET in SYSNOI matches the value of MGATE in PULSED. 2. XMTINO = 1 when MGATET = False and XMTINO = AVGMTI when MGATET = True. RESULT: OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-12	OBJECTIVE: Check that subroutine JAMMER accesses the correct formula for on-board jammer signal MTI attenuation factor. PROCEDURE: 1. Add DATAJAMR to the revised Sample 11 input file of Test 23-8, where IPRJAM = 0 CFREQ = 3000.0 JAMTYP = 1 DBJTOS = 20.0 PWRJAM = 0.01 LADSOJ = 000000 GJAMDB = 5.0 LODSOJ = 000000 BWJAMR = 0.52 ZSOJM = 0.0 DBLOSJ = 3.0 MSLSOJ = 0 2. Run ALARM using VMS debug with the input file of step 1. Set breakpoints in RDRINT, PULSED, JAMMER, and RESMTI. 3. Note the value of AVGMTI at line 556 of RDRINT. 4. Note the value of MGATE at line 433 of PULSED. 5. Note the value of MGATET at line 388 of JAMMER and compare to the value from step 4. 6. Check whether line 392 or line 402 is executed in JAMMER, and note the value of XMTIOB. 7. If RESMTI is called from JAMMER, note the value of the passed parameter (DOPCSCR) in RESMTI and compare with independently calculated value. 8. Set a watch on MGATE, to observe when it changes value in PULSED. 9. Repeat steps 4-8. 10. Successively deposit into variable JAMTYP the values 3, 4, and 5 and repeat steps 4-9 for each value. VERIFY: 1. Value of MGATET in JAMMER always matches value of MGATE in PULSED. 2. XMTIOB = 1 when MGATE is False. 3. If MGATE is True and JAMTYP = 1 or 5, XMTIOB = AVGMTI. 4. If MGATE is True and JAMTYP = 3 or 4, XMTIOB is calculated by calling RESMTI. 5. DOPCSR matches independent calculations in step 7. RESULT: OK
23-13	OBJECTIVE: Check that subroutine JAMMER accesses the correct formula for stand-off jammer signal MTI attenuation factor. PROCEDURE: 1. Add DATAJAMR to the revised Sample 11 input file of Test 23-8, where IPRJAM = 0 DBLOSJ = 3.0 JAMTYP = 2 CFREQ = 1325.0 NUMSOJ = 1 LADSOJ = 504500 PWRJAM = 10.0 LODSOJ = 075000 GJAMDB = 10.0 ZSOJM = 3000.0 BWJAMR = 15 MSLSOJ = 0 2. Use VMS debug to run ALARM with the input file of step 1. Set breakpoints in RDRINT and JAMMER. 3. Note the value of AVGMTI at line 556 of RDRINT. 4. Note the value of MGATEJ at line 647 of JAMMER. 5. Check whether line 659 or line 663 is executed in JAMMER and not the value of XMTISJ. 6. Continue execution. When JAMMER is re-entered, break at line 647 and deposit into variable MGATEJ the opposite value (True or False) from the value in step 4. 7. Repeat step 5. VERIFY: XMTISJ = 1 when MGATEJ = False and XMTISJ = AVGMTI when MGATEJ = True. RESULT: OK

Table 2.23-3 MTI Test Cases

TEST CASE ID	TEST CASE DESCRIPTION
23-14	OBJECTIVE: Check for reading, printing, and error checking of inputs. PROCEDURE: 1. Run ALARM, using Sample 11 as the input file. 2. Examine the printed output for DATARADR and compare with independent assessment of inputs. VERIFY: 1. Printed ALARM output matches independent assessment of inputs. 2. No errors are generated. RESULTS: OK
23-15	OBJECTIVE: Check error-handling in RDRERR. PROCEDURE: 1. Revise the Sample11 input file to include the following errors: NDELAY = 0 FMTIDB = -2.5 SIGMAC = -5.7 RMTIMN(1) = -10.5 RMTIMN(2) = 0.0 RMTIMN(3) = 50.0 RMTIMN(4) = 40.0 RMTIMX(1) = 0.0 RMTIMX(2) = 30.0 RMTIMX(3) = 35.0 RMTIMX(4) = 60.0 AMTIMN(1) = -30.0 AMTIMN(2) = 60.0 AMTIMN(3) = 100.0 AMTIMN(4) = 180.0 AMTIMX(1) = 30.0 AMTIMX(2) = 120.0 AMTIMX(3) = 80.0 AMTIMX(4) = 240.0 Leave other records unchanged. 2. Independently predict error messages that should be generated. 3. Run ALARM, using Sample 8 with revised radar data as input. 4. Compare ALARM generated errors to independent predictions. VERIFY: ALARM errors match predicted errors. RESULT: 1. Errors should have been produced, but were not produced for RMTIMN > RMTIMX or AMTIMN > AMTIMX. 2. The ALARM input guide incorrectly states that NDELAY > 0, when it should be $\geq$ 0.

2.23.5 Conclusions and Recommendations

Code Discrepancies

No major problems or anomalies were found with the ALARM MTI references or the code implementation. The minor problems found in RDRERR could be eliminated by replacing lines 522 and 560 with the following:

Line 522: 'IF (RMTIMX(IGATE) .LT. RMTIMN(IGATE)) THEN '

Line 560: 'IF (AMTIMX(IGATE) .LT. AMTIMN(IGATE)) THEN'

Code Quality and Internal Documentation

Code quality is generally good. Internal documentation is adequate, but should be improved to provide a consistent set of information in the subroutine headers and to correct spelling errors.

External Documentation

External documentation for MTI in ALARM 3.0 is generally good. The manuals have been updated to match the new code in this version. The minor error in the User's Manual could be corrected by replacing "NDELAY > 0" with "NDELAY≥ 0".